REMARKS/ARGUMENTS

Claims 1-30 are pending.

Claims 1 and 6 were rejected under 35 U.SC. § 112, 2nd Paragraph.

Claims 1-30 were rejected under 35 U.S.C. § 102(e) for allegedly being anticipated by Narad et al., U.S. Patent No. 6,701,338.

A telephonic interview was held on Tuesday, March 9, 2005 between Examiners El Chanti and Najjar, the inventor, and the undersigned in connection with related U.S. Application No. 09/538,132, filed March 29, 2000.

The Examiner's attention is drawn to a response filed July 6, 2004, wherein an IDS submission of a PCT reference was filed with the response. A copy of the cited reference had been submitted in an IDS filed at the time of filing the instant application, but apparently was not received in the file wrapper. Notice of that reference is given hereby, and consideration of that reference is respectfully requested.

The present invention is directed to processing of data packets; e.g., data packets in a communication network. A distinguishing aspect of the present invention as recited in claim 1, for example, is "providing a language definition including a grammar" and "processing ... data in accordance with a formal language processing technique using said language definition ...including parsing said network data using said grammar."

Each of the independent claims 1, 6, 11, 18, and 27 recites a language definition and/or a grammar, and processing data packets using a formal language processing technique including parsing.

A close reading of Narad et al. reveals the following additional distinctions from the present invention:

Narad et al. at column 36, line 19 to column 37, line 22: Section IV Classification Engine

Narad et al. describe their Classification Engine (CE) as "a microprogrammed processor designed to accelerate predicate analysis in network infrastructure applications." *Col. 36, lines 20-21.* The components of the CE include:

- header parsing - This involves comparing portions of bit fields in a header packet to determine a match/no-match condition.
- table lookup - This component is a simple table lookup operation to identify a record associated with a packet. The packet is used to perform the table lookup function; e.g. by comparing fields in the packet or by a hashing technique.
- checksum - This is basically summing the bytes in the packet to arrive at a value to determine validity.
- In addition, Narad et al. describe being able to perform computations on arbitrary data fields in a packet.
- The remainder of the cited portion describes data and hardware implementation aspects of the CE.

These foregoing components of the CE neither individually nor collectively constitute a language definition, nor do they constitute a formal language definition including a grammar or a lexical token representation of classification rules. Although Narad et al. disclose "header parsing," they describe this component as performing bit-field comparisons. There is no grammar or parsing according to a grammar in the classification process.

In Narad et al., classification rules are defined in a language called NCL language that performs bit-field comparisons, but the classification rules themselves do not constitute a definition of a formal language with grammar or lexical rules.

Moreover, the CE is designed to perform predicate analysis, which in this case includes comparing portions of bit fields in a header packet to one or more constants to produce a match/no-match result. *Col. 36, lines 26-31.* Comparison of bit fields does not involve a language definition. Moreover, comparing bit fields does not constitute a formal language

Appl. No. 09/557,736
Amdt. sent March 11, 2005
Reply to Office Action of November 17, 2004

PATENT

processing technique as understood by those of ordinary skill in the art of compiler techniques; for example, comparing bit fields does not constitute parsing.

Narad et al. at column 58, lines 55-59: Section VI Programming Model

According to Narad et al. "[t]here are two modules required: the application processor (AP) module, and the policy engine (PE) module. Application code in the AP module runs on the host processor, and is most appropriate for processing not requiring wire-speed access to network frames. Application code for the PE module comprises the set of classification rules written in the NetBoost Classification Language (NCL), and an accompanying set of compiled actions (C or C++ functions/objects)." *Col. 58, lines 55-58.* Narad et al. quite clearly state that "the set of classification rules written in the NetBoost Classification Language (NCL)." The classification rules are executed by the Classification Engine (CE) to perform packet classification. The provided NCL itself does not perform packet classification. By comparison to the present invention, a language definition having a grammar is provided, and it is the language definition *itself* that is applied to packets to perform parsing of the packets.

Narad et al. also discuss the use of "an accompanying set of compiled actions (C or C++ functions/objects)." *Id at line 59.* However, it should be carefully noted that these compiled actions have nothing to do with the classification function. Instead, the compiled actions are associated with the results of the classification process. Narad et al. mention "[a] language interface to describe Classification and to associate Actions with the results of the Classification." *Col. 4, lines 46-47.* It appears that a function of the language interface is to associate actions with classified packets.

Narad et al. at column 59, lines 19-28, column 60, lines 22-27: Section VI Programming Model

According to Narad et al. "[c]lassification rules are specified in the NetBoost Classification Language (NCL) and are compiled on-the-fly by a fast incremental compiler provided by NetBoost. Actions are implemented as relocatable object code provided by the application developer. A dynamic linker/loader included with the NetBoost platform is capable of linking and loading the classification rules with the action implementations and loading these either into the host (software implementation) or hardware PE (hardware implementation) for

execution." *Col. 59, lines 19-28.* "The application programmer specifies rule predicates within an ACE using Boolean operators, packet header fields, constants, set membership queries, and other operations defined in the NetBoost Classification Language (NCL)." *Col. 60, lines 22-26.* The rules are an NCL program. *Id at line 27.* Narad et al. clearly teach that the classification rules are an NCL program and are compiled for execution in the PE hardware (more specifically on the Classification Engine in the PE) to perform packet classification. The provided NCL itself does not perform packet classification.

By contrast, in the present invention, the data packets themselves are processed in accordance with the provided language definition. *Claim 1.* By analogy with the concept of programming languages, a data packet would be analogous to a "program" written in the provided language definition. Further analogizing, the data packets are "compiled" when they are parsed according to the grammar. This, is a fundamental distinction between the present invention and all of the cited art of record including Narad et al.

Narad et al. at column 62, line 25 to column 77: Section VII Classification Language

Narad et al. discuss details of their NetBoost Classification Language (NCL). However, it has been shown that Narad et al. provide NCL which serves to define their classification rules. It is the classification rules which, when compiled, are used to perform packet classification. By contrast, the present invention, provides a language definition which serves to define the data packet *themselves*, and that classification of a data packet occurs when it is "compiled"; i.e., parsed in accordance with the grammar of the provided language definition. Consequently, none of the disclosed specifics of NCL provided in this section of the Narad et al. reference teaches or even suggests the present invention.

Narad et al. at column 77 to column 123: Section VIII ASL

Narad et al. discuss details of their Application Services Language (ASL). The ASL "provides a set of library functions available to action code that are useful for packet processing." *Col. 74, lines 49-51.* The action codes can be implemented in C or C++ programming language. *Id at lines 55-56.* "The ASL is commonly used for manipulating the

Appl. No. 09/557,736
Amdt. sent March 11, 2005
Reply to Office Action of November 17, 2004

PATENT

contents of packets." *Col. 78, lines 61-62.* The ASL also contains macro definitions to aid in debugging and code development. *Col. 79, lines 9-11.*

Narad et al. state that "[w]hen a rule's action portion [code] is invoked <u>because the rule predication portion evaluated true</u>, the action function must return a code indicating how processing should proceed. The action may return a code indicating it has disposed of the frame (ending the classification phase), or it may indicate it did not dispose of the frame." *Col. 79, line 66 to col. 80, line 7 (underlining added).* As indicated by the underlined portion, the action code is invoked after a packet has been classified; i.e., evaluated true. Column 9, lines 1-30 provides a list of some action codes.

The ASL and its action codes do not participate in packet classification, but rather in the processing of a packet, <u>after</u> the packet has been classified.

CONCLUSION

In view of the foregoing, Applicants believe all claims now pending in this Application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested. If the Examiner believes an addition telephone conference would expedite prosecution of this application, the Examiner is invited to contact the undersigned at 650-324-6352.

Respectfully submitted,

George B. F. Yee
Reg. No. 37,478

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: 650-326-2400
Fax: 415-576-0300
GBFY:cmm
60381788 v1